1987

A heuristic algorithm to assign orders into order picking lots in a semi-automated warehouse /

Justin Khoe
Lehigh University

A HEURISTIC ALGORITHM TO ASSIGN ORDERS INTO ORDER PICKING LOTS IN A SEMI-AUTOMATED WAREHOUSE

by

Justin Khoe
(Kian Hoat Khoe)

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Manufacturing Systems Engineering

Lehigh University

October 1987

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Manufacturing Systems Engineering (MSE).

September 18, 1987
Date

George R. Wilson
George R. Wilson
Professor in Charge

Roger N. Nagel
Roger N. Nagel
MSE program Director

F. Erdogan
Fazil Erdogan
Mech. Eng. Dept. Chairman

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. George R. Wilson for his time and patience during the development of this thesis. The completion of this thesis was made possible through his guidance.

Also, special thanks to Dr. Mikell P. Groover for providing me with a research opportunity on his project with IMPACT Technologies, Inc. Their dedication to research and teaching at Lehigh is truly admired. In addition, I would extend my gratitude to Joe Kaczmarek of IMPACT.

Finally, I would like to thank my parents for their love and patience during my master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The scheduling aspect of order picking involves two major areas : (1) grouping orders into picking lots and (2) finding a best sequence of items for picking the lot. Much research has been done in finding the best sequence for picking the lot. This is a classical problem in operations research called the Traveling Salesman Problem(TSP). On the other hand, relatively little research has been done in the area of grouping orders into picking lots. It appears that only three individuals or groups have contributed to this area of research : Elsayed and Stern [9], Barrett [1], and Gillett and Miller [10].

The heuristic developed by Barrett, based upon Eilon [7], is limited to a manual operation and does not consider the locations of items in an order. A heuristic approach developed by Elsayed and Stern is limited to the extent that an order can not be picked partially and must have a total required capacity less than the order picking vehicle.

This paper introduces a heuristic approach by considering the size of items, allowing a partial capacity of an order to be picked, and an order that may have the total required capacity greater than a picking vehicle.

A simulation was run to compare Barrett's algorithm with the proposed algorithm and to study the proposed algorithm under various variables such as pick wave size and number of items in an order.

The simulation study shows that the proposed algorithm has two advantages over Barrett's algorithm with respect to the total required picking lots in a pick wave and order picking utilization. The execution time in CPU seconds of the proposed algorithm is longer than that of Barrett's algorithm. Also, the proposed algorithm requires more CPU seconds and picking lots in any pick wave as the number of orders in a pick wave or number of items in an order increases.

Then, in terms of implementation and application, the proposed algorithm can be easily programmed in any language and applied to other types of order picking systems such as, man-on-board and zone picking.

1. INTRODUCTION

This research was designed to develop and formulate an operating procedure for the assignment of customer orders into an order picking lot. This procedure affects the order picking productivity of a warehouse, subject to warehouse design and customer demand patterns. Thus, an effective operating procedure better ensures that a warehouse can shorten its delivery time and improve its customer service.

In the past decade, technology has been introduced to warehouses (otherwise known as distribution centers). Warehouses had been largely neglected in the discrete or process manufacturing industry. However, the service industry, such as mail order companies, has long recognized a warehouse or a distribution center as a formidable weapon to gain a better market share in its industry.

Researchers and scholars in the area of operations research have also channeled their energy into warehousing and materials handling. Recently, more research has been performed in the area of integrating a warehouse with a production system. Thus, warehousing has become a "hot" topic among today's industrial and manufacturing systems engineers and researchers.

The impact of a warehouse or distribution center on customers is direct and powerful. To gain a market share a company should deliver a saleable product or service while addressing three essential

criteria: fast delivery time, excellent quality, and lower overall cost than that of its competitors. If a company operates, for example, in a made-to-stock environment, only delivery time and distribution cost seems to be addressable issues for an operating order picking procedure.

Both fast delivery time and lower overall cost would definitely contribute to better customer satisfaction. This increase in customer satisfaction can be achieved if warehouse operations have scored high in the following performance criteria:

1. High productivity
2. Efficient design
3. Good control of inventory

This paper is concerned primarily with productivity of an order picking operation in a warehouse. High productivity can be achieved through high throughput, job satisfaction, and fault-tolerant performance. Throughput, the principal measure of productivity, is defined as the number of picks or putaways per unit of time. A pick is defined as a business transaction to get a requested item from its assigned location in a storage system either manually or automatically. Putaway, on the other hand, is defined as a business transaction to put an item into its assigned location in a storage system either manually or automatically. Efficient design considers the effective use of space, equipment, and manpower. With effective design, higher throughput is more achievable. Good control of inventory ensures a high level of accuracy in any inventory

transaction. The accuracy of an inventory record means that the inventory quantity of an item or SKU(Stock Keeping Unit) must represent the "exact" quantity of that item in a storage system. This emphasis on accuracy should lead to a better coordination and interaction not only among internal functional units, but also between a company and its customers.

Efficient design provides the planning framework that allows efficient and effective operational procedures to enhance productivity. For picking, throughput is a function of picking rate. The picking rate can be increased several ways: reducing picking travel time, utilizing the capacity of picking vehicles effectively and efficiently, and designing a good layout of a warehouse to ensure a good flow of materials.

Thus, developing a picking procedure is a very challenging task. Moreover, a picking procedure should be designed specifically for each application. The specification of picking procedure depends on four factors:

1. Objective
2. Equipment
3. Layout of a warehouse
4. Budget

The objective of a picking procedure should be formulated in light of the other factors: equipment, layout, and budget.

Three types of equipment used for order picking are a narrow-aisle storage/retrieval vehicle, a narrow-aisle order picking

vehicle, and a man-on-board AS/RS. The man-on-board AS/RS operates in an aisle of a pallet rack storage system and is used for picking less-than-unit-load items. On the other hand, the narrow-aisle storage/retrieval vehicle is used for picking unit load items. One vehicle operates only along one aisle unless transfer machines are used. Both of these vehicles are known as aisle-captive; i.e., the vehicle operates in a dedicated fashion along one aisle. In contrast, the narrow-aisle order picker truck can transfer from one aisle to another. The size of items for an order picker truck is limited by what can be handled manually in a safe manner. With this in mind, an order picking procedure should be suited to the equipment used.

The layout of a warehouse may be one of a number of designs; for example, a one level storage system, a multi-level storage system, or a storage system with a conveyor. In addition, a storage system can have a layout in block type (e.g., grocery warehouse), or pallet rack (e.g., consumer products--non-food warehouse). With these alternatives of layout types, an order picking procedure should be designed specifically to match the layout.

Finally, each procedure should be designed within an allocated time and budget. Obviously, without this consideration, any procedure can not be successfully implemented within available resources and technology.

Order picking can be accomplished in several ways: manual picking, semi-automated, and automated. Manual order picking is done by a worker who walks along and/or across an aisle. A computer system

to generate picking assignments and reports may be used in this method. In a semi-automated order picking system, both a mechanical transporter and a worker carried by the transporter are used to retrieve and store SKUs in the storage system. The manual and semi-automated order picking systems are sometimes referred to as Man-to-Part order picking systems. An automated order picking, commonly called Part-to-Man order picking, is done without a worker in the aisle and it is typically done aisle by aisle. Part-to-Man order picking is exemplified by an AS/RS.

In the next chapter, a review of current research in the area of order picking and related issues are presented. Based upon this literature review, an assignment of customer orders into a picking lot in a man-to-part or part-to-man order picking model will be proposed. Finally, an analysis of the proposed model compared to existing approaches and its possible extensions for future research will be discussed.

2. LITERATURE REVIEW

Management of a warehouse faces two areas of concern when scheduling an order picking operation: (1) Which orders are selected to be picked among a set of orders and (2) What picking sequence or procedure should be followed to get all items of selected orders from their storage locations. Appropriate answers to these two questions are key to the design of an operating policy or procedure for an order picking system. And, as mentioned earlier, an order picking procedure should be developed with a particular order picking hardware in mind. To prevent reinventing the wheel, an extensive review of literature related to the order picking issue was conducted. Three areas of literature of particular interest are in-the-aisle order picking, the traveling salesman problem (TSP), and assignment of orders into a picking lot. Research on in-the-aisle order picking provides information on algorithms that are used to sequence a picking activity, distance metrics that are used to represent different types of order picking vehicles, and several approaches to evaluate an order picking procedure. The TSP literature reveals more general algorithms to sequence a tour while minimizing a distance traveled or a travel cost. The last area of the review provides information on what rules have already been developed to select, group, and sequence orders to be picked. In the following section those three areas of research are explored in detail.

2.1. In-the-Aisle Order Picking

Hugo Mayer Jr. [17], a Western Electric engineer, appears to have done the earliest work in a non-automated warehouse. He proposed three models for three different storage and retrieval methods: order picker, mechanical transporter, and fork-lift truck. In his models the order picker is defined as a picker or warehouse worker who walks to pick or retrieve items from a one aisle, one-level storage system. The mechanical transporter is represented by a storage and retrieval (S/R) machine operated manually to pick or retrieve items along one aisle of a storage system. On the other hand, the fork-lift truck is operated by a worker and its working area is not restricted to one aisle. Based upon each of these storage and retrieval methods, Mayer formulated a single and a dual cycle time of operation under a rectilinear distance metric. His definitions of a single and a dual cycle of operation are as follows:

> A single cycle of operation is defined as either depositing a unit load in storage or retrieving a unit load from storage.
> A dual cycle of operation is defined as both depositing a unit load in storage and retrieving it from storage, where deposit specifically precedes retrieval.

Because of advances in technology, a mechanical transporter such as an Automated Storage and Retrieval System (AS/RS) operates differently. Thus, Mayer's formulation based upon the first two models, order picker and mechanical transporter, may not be accurate and appropiate for AS/RS. Research has shown that the traveling time in the AS/RS should be modelled in the Chebyshev metric rather than

the rectilinear metric. The Chebyshev metric is appropiate for the AS/RS because two separate motors power the S/R machine. The two motors work simultaneously in horizontal and vertical directions when the S/R machine moves from point A to point B, as depicted in Figure 1. Hence, the travel time between point A to point B is equal to the maximum of the horizontal or vertical travel time. On the other hand, the travel time in the rectilinear metric is equal to the sum of the horizontal and vertical travel times, as depicted in Figure 2.

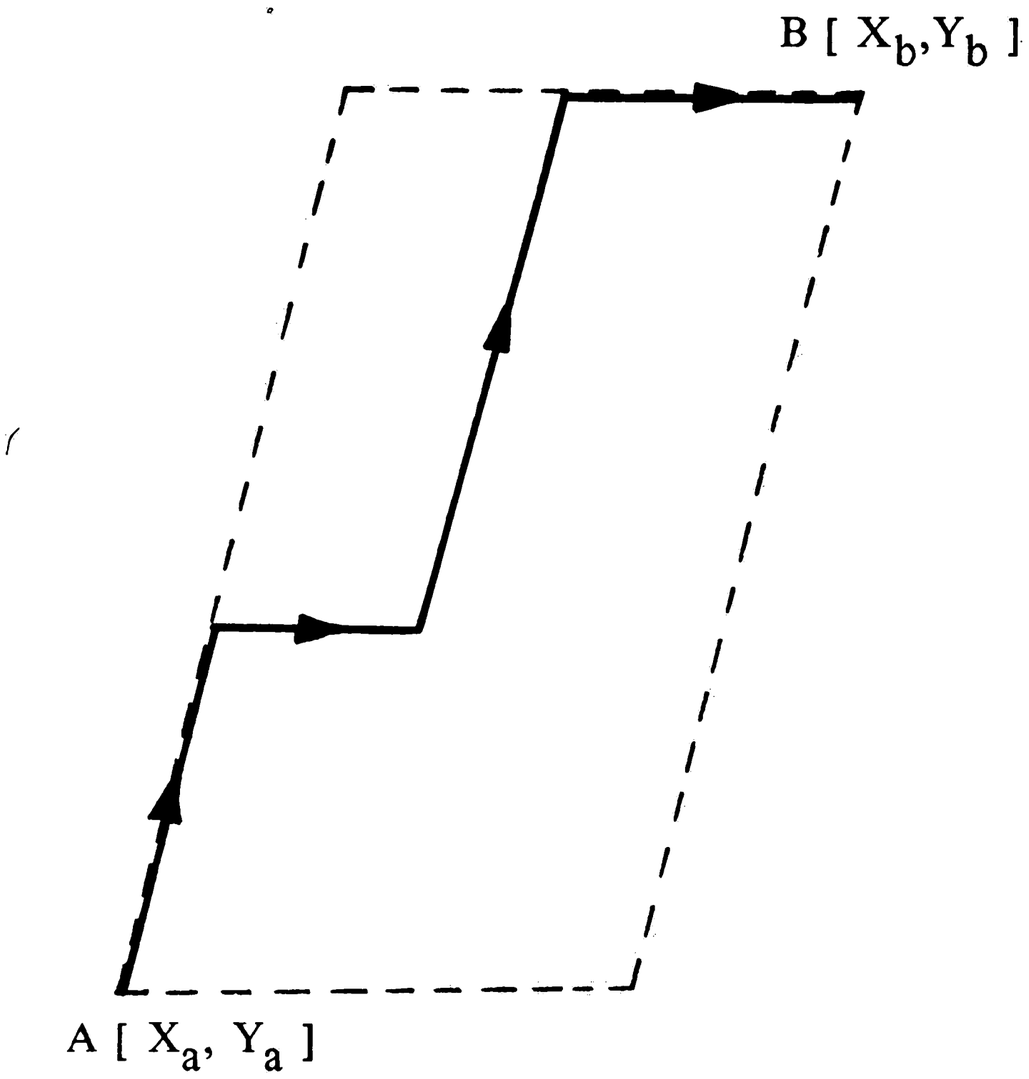
Hausman, et al [14] performed a comparative study of three storage assignment policies: random, full turnover, and class-based turnover in an automated warehousing system. Their study used continuous variable analytical models and discrete variable evaluation procedures to represent rack storage locations. The full-turnover assignment policy is defined as having the highest turnover pallet assigned to the location closest to I/O. To determine the frequency for a turnover assignment, an ABC distribution was used. The class-based assignment policy is defined as partitioning a pallet rack into a small number of classes and within each class pallets are assigned randomly. The study found that S/R travel time on a full-turnover based storage policy is less than travel time based on a random storage policy.

Later in a different article, Hausman, et al [12] extend the previous work by combining both storage and retrieval transactions, commonly called an interleaving policy. Several interleaving policies were compared in their study with respect to expected S/R round-trip

travel time(i.e., completing one store and one retrieve with starting and final destination the I/O point).

Both of these studies assumed a deterministic environment; so, in order to learn more about the dynamic behaviour of a warehouse (i.e., storage and retrieval queueing), they did a simulation study of their previously proposed analytical models. A reader interested in the result of their simulation study is referred to their paper [22]. The primary result of this study is that the previously developed deterministic analytical models can be used in a stochastic setting, with a few restrictions.

Bozer and White [6] formulated the expected travel times for a single and dual cycle of operation under the Chebychev metric. Their formulation is based on randomized storage policy, a continuous rack storage system, and several Input/Output (I/O) location alternatives in a rack storage system. Their analysis shows that a travel time formulation based upon a continuous function representing a discrete rack performs in a satisfactory fashion with the largest percentage deviation reported to be 0.2506 % from one based upon a discrete function.



t_{AB} = Travel time from A to B

t_{AB} = $MAX[(X_b-X_a)/V_h, (Y_b-Y_a)/V_v]$

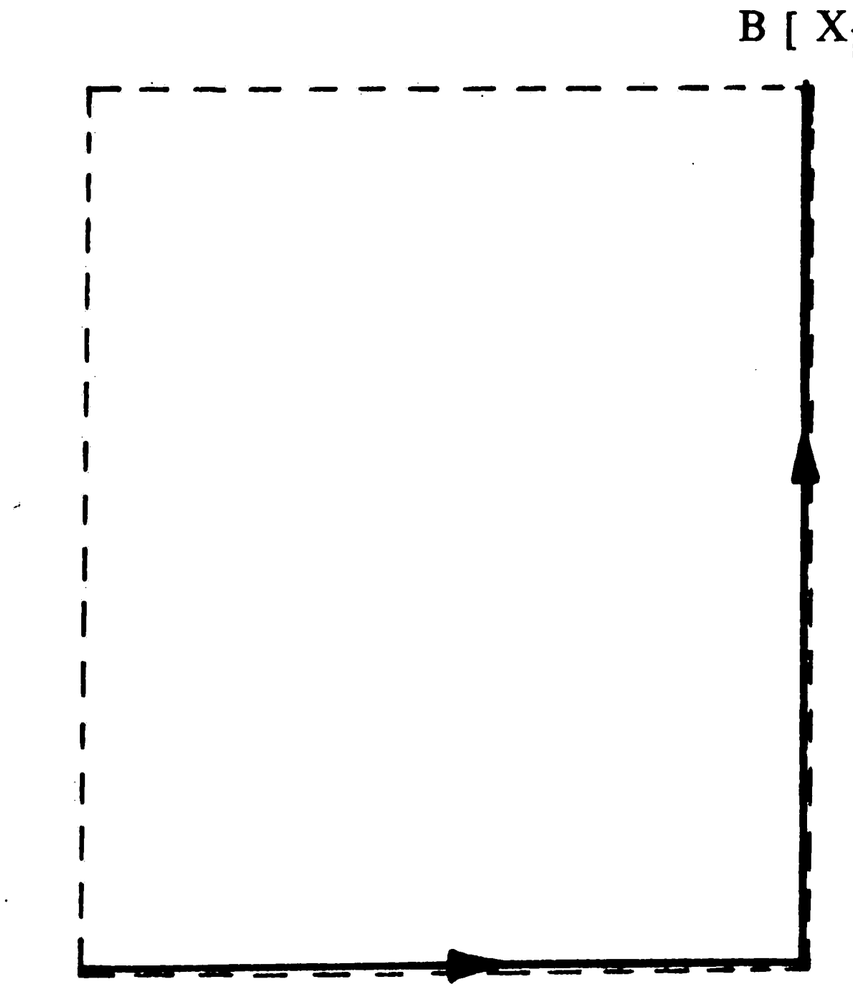
X_a,Y_a = Coordinate of point A.

X_b,Y_b = Coordinate of point B.

V_h = Horizontal motor speed.

V_v = Vertical motor speed.

Figure 1. Travel Time in Chebyshev metric



$$t_{AB} = (X_b - X_a)/V_h + (Y_b - Y_a)/V_v$$

Note : same variable definitions as used in Figure 1.

Figure 2. Travel Time in Rectilinear Metric

2.2. Traveling Salesman Problem

The Traveling Salesman Problem(TSP) is a classical problem in Operations Research. This problem can be solved either by an exact or a heuristic approach. An exact algorithm always gives an optimum solution; however, the computation time increases exponentially with the number of locations to be visited. A heuristic algorithm, on the other hand, gives an optimum or near-optimum solution in reasonable time, but an optimum solution can not be guaranteed. Hence, it can handle more locations than the exact algorithm in terms of computational time. In addition, Papadimitriou [18] has shown that TSP is NP(Non-Polynomial)-complete based upon a set of points on either the Euclidean plane or the Rectilinear plane. Much research has been done in this area; however, the established TSP algorithms are mostly applied to the vehicle-dispatch problem.

Lin and Kernighan [16] introduced two TSP algorithms using the Euclidean metric. The first algorithm guarantees an optimal solution for up to 13 locations. However, the second algorithm only provides a locally optimal solution for up to 145 locations. Lin and Kernighan also present a procedure to achieve an n-optimal tour, where n = 1, 2, 3, ...k (k = integer). The n-optimal tour is defined as a tour that is still optimal even though any n links that are part of the tour are replaced by any n other links. For those interested in further reading, a review and comparison of several TSP heuristics based upon the Euclidean metric has been done by Golden, et al [11].

For a review of results associated with various exact and "matrix driven" heuristic procedures the reader is referred to Parker and Rardin [19]. The TSP procedures for the Rectilinear metric and Chebyshev metric have received relatively minor attention. These two metrics, however, are more appropiate to paths associated with the picking sequence of an order picking operation. Therefore, some TSP heuristic algorithms which are based upon these two metrics will be discussed next.

As previously mentioned, the order picking method is dependent of the layout of a warehouse. Geometric considerations derived from the layout have been used to develop more efficient TSP heuristic algorithms. Ratliff and Rosenthal [20] developed the TSP heuristic algorithm based upon graph theory for a rectangular warehouse having a rack storage system. For their algorithm, the distance between any points in a warehouse is based upon a rectilinear metric. Ratliff and Rosenthal presented a heuristic to sequence order picking in a rectangular warehouse rather than just in an aisle. The computational time for their heuristic is linear in the number of aisles--not number of picks. They reported that it took about one minute to solve a 50-aisle problem using an experimental computer program written in BASIC for an Apple III microcomputer. However, they did not mention the applicability and extensibility of their heuristic for order picking involving picks located at a rack level higher than the first rack level.

Bartholdi and Platzman [2] developed a TSP heuristic algorithm

using the concept of a spacefilling curve. They claimed their heuristic has four attractive features. First, the heuristic basically requires only a sorting routine once the spacefilling curve is defined for a layout of a warehouse. Second, the heuristic, hence, works very fast, depending on the sorting scheme used. Third, it is applicable to a dynamic environment because adding and deleting data from the sorting routine is simple and fast. Therefore, no lengthy recomputation is required as for some other TSP heuristics such as, Ratliff and Rosenthal [20]; Lin and Kernighan [16]; and Held and Karp [15]. Fourth, from an implementation point of view, the heuristic is trivial to code. However, the heuristic does not yield a tour length or travel time. They have used this heuristic to solve a planar traveling salesman problem and to design efficient bin-numbering schemes. In their paper, they concluded that their heuristic will work for any structure of warehouse. It would, however, be quite interesting to see it applied to a rectangular warehouse, having a rack storage system.

Bozer, et al [5] evaluate several heuristic procedures based upon geometric approaches utilizing the Chebyshev metric. These procedures sequence man-to-part order picking in one aisle. Their primary focus is the trade-off in tour quality vs run time. Through empirical analysis, they found that the 1/2 band insertion heuristic gives a more consistent result in terms of tour quality and run time with various arrangement of storage locations in an aisle. The 1/2 band insertion heuristic groups pick locations into two regions:

blocked and unblocked. Pick locations in the unblocked region are sorted, using their horizontal coordinate to form a partial tour. On the other hand, pick locations in the blocked region are inserted into the partial tour, using a minimum cost insertion scheme. The procedure is also conceptually simpler and easier to implement than other algorithms in their study. Also, they reported that the 1/2 band heuristic without the insertion technique has been used widely in industry.

This literature review indirectly contributes to the development of the proposed algorithm. The proposed algorithm is only concerned with a "loading" problem. However, this review is definitely useful in analyzing the throughput of an order picking vehicle because, by using one of the TSP methods, the travel time can be determined. Solving the TSP problem is a scheduling implementation of the loading plan given by the proposed algorithm. The TSP implementation would be a useful extension of the present work.

2.3. Assignment of Orders into a Picking Lot

Gillett and Miller [10] have formulated a new procedure to solve medium as well as large scale vehicle-dispatch problems. They decomposed a given set of locations into several routes. To form each route they proposed a sweep algorithm subject to the weight and distance constraints of the vehicle. This sweep algorithm is based upon the polar coordinate of the locations to be visited with respect to an arbitrary point. Then the path of this route is solved by using

the heuristic algorithm for TSP developed by Lin and Kernighan. Lin and Kernighan's algorithm was selected because it used less computer time than Held and Karp's algorithm which is based upon the branch and bound technique. Gillett and Miller's analysis has produced the interesting result that the number of locations per route has the greatest effect on total computational time. In other words, the amount of computation required increases linearly with the number of locations if the average number of locations per route remains relatively constant. Even though their study does not address the order picking issue, their approach provides some insights into the problem of order picking lot assignment and sequencing an order picking lot in an aisle.

In order to consolidate some of the orders in any pick wave (defined as a set of orders which have the same destination; which are loaded into the same truck; or which have the same shipping time and due date), Barrett [1] has proposed several methods that are amenable to manual operations provided the number of orders are small.

Barrett introduced four methods for assigning customer orders to a picking lot at any one time. His formulation assumed a full case order picking and each order has quantities less than or equal to the capacity of an order picking vehicle. The procedures were analyzed in terms of four performance measures:

1. Number of picking lots
2. Minimum number of lots

3. Number of stops

4. Travel time

The minimum number of lots is a lower-bound, indicating a minimum number of picking lots formed in a pick wave, which is greater than or equal to the required total capacity of orders in a pick wave. The LOAD algorithm, derived from Eilon [7], produces fewer lots compared with the other algorithms. However, this benefit is gained only if the number of orders in any pick wave is more than five. Also, in terms of percent of minimum number of lots, the LOAD algorithm shows a more consistent result even though the pick wave size increases. Since the LOAD procedure produces fewer lots, it has more stops. Barrett's analysis shows that the pick wave size does not affect the number of stops. The travel time, calculated using a 1/2 band heuristic proposed by Gudehus [13], decreases slightly when the pick wave size increases. However, the travel time decreases significantly if a random storage policy is substituted for a turnover-based storage policy. Hausman, et al [14], reported the same result. Barrett introduced the concept of fractional value, defined as the reciprocal of the maximum number of an item or SKU that is required to fill a pallet, tote, box or bag. This concept provides a good approximation for a pallet or box filled with items or SKUs having different sizes. All algorithms proposed by Barrett contain a similar sorting procedure. Thus, his algorithms do not consider the locations of items belonging to an order. This may result, depending on the data, in a higher travel time for a picking lot.

Elsayed and Stern [9] developed a set of criteria to combine orders into picking lots. They assume that all items have the same dimensional size and shape, and each order must be completed in one tour. Hence, the total capacity of an order must be less than or equal to the capacity of the order picking vehicle. Their procedure has three major steps: finding seed order, order congruency, and addition of orders. In the first step, a seed order is selected using four simple rules associated with either total quantities or total number of locations in an order. Then, based upon this order, another order is selected using three congruency rules. Congruency means that one order can be combined with another order by using predefined attributes existing in both orders. Finally, the seed order may be updated by adding the congruent order to it or the seed order remains the same until the capacity of the order picking vehicle is filled. They reported their proposed procedure is data-dependent. Thus, all possible algorithms derived from the given rules should be used to yield a best possible solution. However, a certain warehouse may follow a specific pattern structure for its orders. Hence, some of the possible algorithms could provide consistently better solutions.

Later, Elsayed [8] performed an extensive Monte Carlo simulation study of the set of rules designed by Elsayed and Stern [7]. The simulation experiment was based upon an automated storage and retrieval system. This time, he reported that the seed rules based upon the maximum number of locations and maximum total number of items among orders in any pick wave are superior to the seed rules

based on the minimum number of locations and minimum total number of items among orders in any pick wave. Also, the congruency rule based upon the maximum number of common locations existing between both a seed order and a candidate congruent order produces better solutions than its counterpart congruency rules.

Scheduling an order picking operation involves two tasks:

1. Group orders into picking lots.

   Three papers contribute considerably in this area (Gillett and Miller [10], Barrett [1], and Elsayed, et al [8,9]).

2. Sequence a tour for a picking lot.

   Four papers give in depth information in this area (Ratliff and Rosenthal [20], Bartholdi and Platzman [2], Gudehus [13], and Bozer et al [5] who did an extensive evaluation of heuristic algorithms used for in-the-aisle order picking).

Also, Hausman, et al [12,14,22] performed an extensive study of storage assignment policies for an automated storage and retrieval system.

3. MODELLING APPROACH

Based upon the literature review, the author found that Barrett's method is fast and easy to implement; however, the distance traveled based upon Barrett's procedure may not always be minimum because his procedure does not consider locations of SKUs to be picked. His procedure is also less applicable to zone picking. Zone picking requires that locations of items making up an order be divided into several zones, according to predefined zone areas, and several pickers are assigned to each zone to complete an order picking.

In contrast to Barrett, Elsayed, et al formulated several rules to assign orders into picking lots. Their formulation considers the locations and quantities of an order. However, their heuristic still requires that an order be picked within one tour or picking lot. In addition, their model assumes that all items have the same volume and weight. It, therefore, appears that the heuristic is somewhat restricted in a practical sense.

Thus, one goal of this research is to modify Elsayed's procedure by considering the volume and weight of each item and also allowing an order to have a total capacity requirement of an order that is more than a picking vehicle possesses.

The following section will be devoted to defining the environment in which the proposed model is evaluated. Then, a set of assumptions for the model is listed. Finally, the model formulation

is set forth.

3.1. Physical description of warehouse and picking vehicle

A warehouse, for the purpose of this study, has two storage areas: (1) reserve and (2) forward picking. Items stored in the reserve area are usually packed in a pallet. These loads are used to replenish items/loads stored in the forward picking area. SKUs of each order are picked from this area. The reader wishing to further study arguments for requiring that forward picking and reserve areas use two distinct scheduling routines is referred to a paper written by Sand [21].

A pallet rack system is used in the forward picking area; whereas, in the reserve area either a pallet rack or block type storage is used.

Items/SKUs retrieved from the forward picking area are in the form of a full case. A full case, in this study, is defined as an individual item which has an integer multiple making up a pallet load. Moreover, the full case may contain either one homogeneous item or several homogeneous items.

Now, let us look at what picking equipment would be appropiate in the foregoing warehouse. Obviously, picking equipment will be used in the forward picking area. Since picking equipment in this study must be able to transfer from one aisle to another, an order picking vehicle or truck would be appropriate. This vehicle can be wire guided or rail guided along an aisle. Both types of guidance will

provide efficiency and stability of vehicle operation along each aisle. The vehicle is controlled manually and its maximum workable height would restrict the pallet rack height. The proposed heuristic model, by no means, is restricted to the foregoing physical description. It could be applied to AS/RS or man-on-board order picking. However, locations of items making up an order may have to be grouped into several aisles, accordingly.

3.2. Heuristic Algorithm for assigning orders into a picking lot

The heuristic algorithm is presented in Pascal-like pseudocode and supplemented with a brief description for each module of the algorithm. Prior to discussion of the algorithm, assumptions under which the heuristic is developed are listed:.

1. The scheduling of a replenishment of items in a pallet rack is not considered.

2. No weight precedence constraint is assumed. In other words, regardless of the arrangement of picked items on the picking vehicle, no crushing occurs.

3. A picking tour starts and ends at the same location. In this study, this location would be a shipping line feeding a shipping dock.

4. The total capacity requirement of picked items in any tour or picking lot is less than or equal to the maximum capacity of an order picking vehicle.

5. All items in a full case can be handled manually in a safe manner.

6. Arrival times of orders are determinitic.

3.2.1. Variables notation and definition :

b = base index for Order Congruency operation.

counter = an index counter for a looping structure unless otherwise specified.

i = seed index, determined through the FINDING-SEED-ORDER procedure.

np = starting index number when loading an order picking vehicle.

x = congruency index.

y = maximum number of items to be picked in any picking lot.

w = an index for a time window or a pick wave where w = 1, 2, 3,.... m.

C = normalized value of the maximum capacity of an order picking vehicle

EL = the remaining capacity of an order picking vehicle.

FV_r = fractional value associated with an item number r

ID_{km} = item number that is associated with the mth member of an order O_k.

LOC_r = the assigned location for an SKU having an item number r in a storage system.

$MATCH_k$ = total amount of ID that exists in both O_b and O_k.

MI_z = congruency index candidate set having z elements

$OCAP_k$ = total load capacity required by an order O_k.

$OSET_w$ = a set of orders in a time window or pick wave w.

PL_{jn} = location number associated with nth member in a picking lot j.

PQ_{jn} = quantity associated with nth member in a picking lot j.

PS_{jn} = SKU number associated with nth member in a picking lot j

QI_j = quantity associated with an item number that can be assigned to picking lot j.

QTY_{km} = quantity that is associated with the mth member of an order O_k.

RN = an index which indicates the first of a number of items in an ordered set O_k that are assigned a zero value

SN = an index which indicates the first item number of an order O_k that has not been assigned into a picking lot.

SNS_j = the sum of a number of SKUs from more than one order in a picking lot j.

$TNIO_k$ = total number of items or SKUs in an order O_k.

$TNIL_j$ = total number of items or SKUs in a picking lot j.

$UTIL\%_j$ = percentage utilization of an order picking vehicle by picking lot j.

$VCAP_j$ = required vehicle capacity of a picking lot j.

NOTE : $r = ID_{km}$

3.2.2. SEED ORDER SELECTION

Elsayed and Stern [9] formulated four alternative rules which can be used to find a seed order. A seed order is an order that will be first assigned to a picking lot and also will be used as a basis for an order congruency operation(picking subsequent orders). The four rules in their formulation are given as follows:

rule 1 : $O_i = MAX\{ TNIO_k : k \in OSET_w \}$

rule2 : $O_i = MIN\{ TNIO_k : k \in OSET_w \}$

rule 3 : $O_i = MAX\{ OCAP_k : k \in OSET_w \}$

rule 4 : $O_i = MIN\{ OCAP_k : k \in OSET_w \}$

Their results, and also that of Elsayed [8], show that rule 1 and rule 3 have a better chance of yielding a near-optimal solution over widely varying data. However, it was not mentioned what happens if more than one order have the same TNI_k or $OCAP_k$. To resolve this case, this author proposes that a seed order is determined through a three step operation. First, a seed order is selected using rule 1. If the result yields more than one candidate order for a seed order, a second step should be executed. A second step uses rule 3 to select a seed order. If there are still more than one candidate order, the first order encountered having maximum OCAP is selected to be a seed order. Figure 3 gives the pseudocode for the seed order selection procedure.

******** procedure FINDING-SEED-ORDER ********

NOTE : based upon rule 1 formulated in ELsayed [9].

step 1 : Find $X^* = \{x \mid MAX [TNIO_k], \forall k \in OSET_w\}$
step 2 : If a tie occurs then do

Find $Y^* = \{y \mid MAX[\ OCAP_y\ ],\ \forall\ y \in X^*\}$

step 3 : If a tie occurs again then do
Select the first element in Y^*.

****** END of FINDING-SEED-ORDER **********

Figure 3. Procedure for Finding-Seed-Order

3.2.3. ORDER CONGRUENCY DETERMINATION

To find an order that is congruent with a seed order, Elsayed, et al [9] formulated three rules.

rule 1 :

$MAX\{\ CARDINAL\ (ID_{km} \cap ID_{im}),\ \forall\ O_k \in OSET_w\ \}$

rule 2 :

$MAX\{\ [CARDINAL\ (ID_{km} \cap ID_{im}) + CARDINAL\ (ID_{km} \cup ID_{im}],\ \forall\ O_k \in OSET_w\ \}$

rule 3:

$$MIN\{\ \sum_{y \in ID_{km}} MIN(d_{yz} : z \in ID_{im}),\ \forall\ O_k \in OSET_w\ \}.$$

According to the simulation result in Elsayed [8], congruency rule 1 produces a much higher number of solutions with minimum travel time. His results show that an operating policy, containing congruency rule 1, rule, 2 or rule 3, produces a solution with the minimum travel time 6518, 3542, or 3485 times out of 12,000, respectively. Thus, it is obvious that order congruency rule 1 is much superior to the other two rules and will be used in this study.

As in the formulation for finding a seed order, a step necessary in case of a tie was not given. Therefore, in case of a tie, choose the first order selected by rule 1.

The procedure for order-congruency determination is described in Figure 4. Based upon a base index, a number of common item numbers existing in both "base" order and each order in a set of orders is calculated. The order that is congruent with the "base" order has a maximum number of intersecting (common) item numbers. However, if more than one order has a maximum number of intersecting item numbers, the first order that happens to yield it is selected as a congruent order.

******** procedure ORDER-CONGRUENCY ********

NOTE : based upon rule 1 formulated in Elsayed, et al [8].

```
z := 0
While  k ∈ OSET_w, except k = b
 do begin

        MATCH_k := 0
        For counter := 1 to TNIO_b
         do begin
               For m := 1 to TNIO_k
               do begin
                     If ID_km = ID_b(counter)  then do
                     MATCH_k = MATCH_k + 1
               end
          end
      If MATCH_k > 0 then
            begin
               z = z + 1
               MI_z = k
            end

end
If z = 0 then do FINDING-SEED-ORDER
Else
          If z >= 1 then
```

```
            begin*
                X  = { x | MAX[MATCH_x] }
                If a tie occurs, select the first element in X*.
            end
        end

******  END of ORDER-CONGRUENCY  ************************************
```

Figure 4. Procedure for Order-Congruency Determination

3.2.4. LOADING A PICKING VEHICLE

Procedure LOADING assigns items of an order(s) into a picking lot. In Elsayed's study, he assumed that all items have the same dimensional size. However, this is a rare occurence in practice. Therefore, to account for various sizes of items, a fractional value concept, introduced by Bruce Barrett [1], is used. Thus, every item in a warehouse has its corresponding fractional value. This value aids in determining the utilized capacity of an order picking vehicle in a picking lot.

Figure 5 will describe a procedure for loading items of orders into an order picking truck. An initialization of three values, n(starting index number for loading), VCAP, and SNS, is required. Items are loaded onto a picking lot one at a time until either the maximum capacity of an order picking vehicle is reached or all items have been accommodated.

```
******  procedure LOADING ******
The values of j, k, np, and VCAP_j will be given through
    corresponding arguments in the CALL statement.
working with a new order
    begin
        n          = 1
```

```
        np          = n
        VCAP_j      = 0
        SNS_j       = TNIO_k
    end

working with load remaining from an order that has been
partially assigned in the previous tour
    begin
        n           = np
        SNS_j       = TNIO_k + (np - 1)
    end

    While (VCAP_j < C) and (n <= SNS_j)
    do begin
        If working with a new order, m = n
        If working with load remaining from an order that has been
        partially assigned in the previous tour, m = n - (np - 1)
        PS_jn       = ID_km
        PL_jn       = LOC_r
        PQ_jn       = QTY_km
        VCAP_j      = VCAP_j + QTY_km * FV_r

        n = n + 1
    end

******  END of LOADING  **********************************************
```

Figure 5. Procedure for LOADING

3.2.5. CAPACITY ADJUSTMENT IN A PICKING LOT

Procedure ADJUST-LOT adjusts an item's quantity in an order that is accommodated only partially in a picking lot. Then, its remaining quantity and other items in an order will be assigned to the next immediate picking lot.

The procedure for ADJUST-LOT is given in Figure 6. This procedure attempts to assign as many items as possible into a current picking lot, if capacity on the order picking vehicle is still available. If the item number can be assigned, at least partially, a new value for

current total load of the vehicle and an adjustment of the corresponding quantity of that item number are performed. Reindexing the sequential item numbers of an order is executed prior to procedure LOADING. Also, two values, picking lot number and vehicle capacity, need to be reinitialized.

```
******  procedure ADJUST-LOT  ******
       the values of j, k, and n will be passed through corresponding
              arguments in the CALL statement.
       m    = SN
       EL = C - VCAP_j
       QI   =TRUNC [EL/FV_r],   where TRUNC is a function that gives the
       result in an integer value that is rounded down.

       If QI > 0 then
       begin
        PS_jn = ID_km

        PL_jn = LOC_r

        PQ_jn = QI
       end

       If QI < 0 then Return to the main program
       VCAP_j    = VCAP_j + QI * FV_r

       UTIL%     = VCAP_j * 100

       QTY_km    = QTY_km - QI

       Call REINDEXING procedure
       {  Initialization for LOADING procedure  }

       j         = j + 1
       VCAP_j    = 0
******   END of ADJUST-LOT  ********************************************
```

Figure 6. Procedure for Adjusting a Picking Lot

3.2.6. REINDEXING AN ORDER

Since all items in an order are sequentially indexed and accessed during any procedure, a partial accommodation of these items into a picking lot requires a renumbering operation. Procedure REINDEXING performs a renumbering operation and is shown in Figure 7. An item number indexed by SN is renumbered as 1. And then the following items are numbered sequentially until the total number of remaining items in an order is reached. Finally, a new total number of items in that particular order is calculated.

```
******  procedure REINDEXING  ******
        The values of K and SN will be passed through
                corresponding arguments in the CALL statement.
        For m = SN to TNIO_k
        do begin
             ID_k((m - SN) + 1)        = ID_km

             QTY_k((m - SN) + 1)       = QTY_km
        end

        RN = (TNIO_k - SN) + 2
        For m = RN to TNIO_k
        do begin
             ID_km = 0;
             QTY_km = 0;
        end

        TNIO_k = RN -1

******  END of REINDEXING  **********************************************
```

Figure 7. Procedure for Reindexing an Order

3.2.7. REARRANGE NOT-YET ASSIGNED ORDER

Procedure REARRANGE shown in Figure 8 performs four tasks. First, a renumbering operation for an order is performed using the REINDEXING procedure. Second, it sums the quantity of all remaining items for an order. Third, this sum will provide an idea of whether or not the remaining items could be assigned into the next picking lot. If the total capacity of the order is less than the capacity of an order picking vehicle, its order number can be removed from a set of orders. Finally, an initial value of the current capacity of the vehicle for the next picking lot is assigned a zero value.

****** **procedure REARRANGE** ******

The values of j, k, VCAP, and RN will be passed through
arguments in the CALL statement.
Call REINDEXING procedure
j = j + 1
$OCAP_j = 0$
{ sum the remaining items in an order to check whether its total remaining capacity can be assigned to the next picking lot.}

For m = 1 to (RN - 1)
do begin
$OCAP_j = OCAP_j + QTY_{km} * FV_{ID_{km}}$
end

If $OCAP_j < C$ then do $OSET_w = OSET_w - \{k\}$

{ Initialization for LOADING procedure }

$OCAP_j = 0$

****** END of REARRANGE **********

Figure 8. Procedure for REARRANGE

3.2.8. MAIN PROGRAM

The main program for the proposed algorithm is divided into four cases and contains all the foregoing procedures. The cases are categorized with respect to two parameters: (1) $VCAP_j$, the current required capacity of a picking lot j compared to C, the maximum capacity of an order picking vehicle and (2) n, the total number of items that have been assigned into a picking lot j compared to SNS_j, the total number of items that are scheduled to be accommodated in a picking lot j.

Figure 9 shows the outline of the main program. The main program is executed if a set of orders exists. The first order to be picked is determined by procedure FINDING-SEED-ORDER and its items are loaded using procedure LOADING. The result of the execution of LOADING will fall within four cases. Case 1 is executed if the required capacity of an order is less than the maximum capacity of the order picking vehicle and also n is greater than SNS_j. Its order number is removed from the set of orders. However, if at this time the set of orders is a null set, then the program is terminated. Since the vehicle still has an extra capacity, a new order will be selected using procedure Order-Congruency. Then, its items are loaded using procedure LOADING.

Case 2 is executed if the required capacity of an order is more than the maximum capacity of the order picking vehicle. Depending upon whether a new order is loaded or a remaining load from an order in the previous tour is loaded, an index to an item number in a particular

order is calculated. The required capacity of the picking lot j is reduced by the total capacity of the particular item. Then procedure ADJUSTLOT and LOADING are called.

Case 3 is executed if the required capacity of an order is less than or equal to the maximum capacity of the order picking vehicle and also n is less or equal to SNS_j. Similar to Case 2, an index to an item number in a particular order is calculated. Then procedures REARRANGE and LOADING are called.

Case 4 represents an ideal case where the required capacity of an order is exactly equal to the maximum capacity of the order picking vehicle. Similar to case 1, its order number is removed from a set of orders. Should the set be a null set, the program is terminated. If not, a new order, determined through procedure FINDING-SEED-ORDER, is assigned to the next picking lot.

The execution of the program will jump from one case to another until all orders are assigned into picking lots.

****** MAIN PROGRAM ******

Call FINDING-SEED-ORDER procedure to obtain the value of i
k = i
j = 1
w = 1
While $OSET_w \neq \emptyset$
do begin
    Call procedure LOADING
CASE 1 : an order has been assigned Completely to a picking lot

    If $VCAP_j < C$ and $n > SNS_j$ then
    begin
        b = k
        $OSET_w = OSET_w - \{b\}$

If $OSET_w = \emptyset$ then do get out of While loop
Call ORDER-CONGRUENCY procedure to obtain
the value of x
k = $\hat{x}$
np = n
Call LOADING procedure
end

****** END of CASE 1: **********

CASE 2 : the Capacity of an order picking vehicle is exceeded

If $VCAP_j > C$ then
begin
n = n - 1
If working with a new order, m = n
If working with load remaining from an order that has been partially assigned in the previous tour, then
m = n - (np - 1)
SN = m
$VCAP_j = VCAP_j - QTY_{km} * FV_r$

Call ADJUSTLOT procedure
np = 1
Call LOADING procedure
end

****** End of CASE 2: **********

CASE 3 : the Capacity of an order picking vehicle is "exactly" reached

If $VCAP_j <= C$ and $n <= SNS_j$ then
begin
If working with a new order, m = n
If workin with load remaining from an order that has been partially assigned in the previous tour, then
m = n - (np - 1)
$RN = (TNIO_k - m) + 2$
Call REARRANGE procedure
np = 1
Call LOADING procedure
end

****** END of CASE 3: **********

CASE 4 : the Capacity of an order picking vehicle has been "exactly" reached and also an order has been Completely assigned to a picking lot ("ideal" condition for any tour).

If $VCAP_j = C$ and $n > SNS_j$ then
begin
    $n = n - 1$
    $OSET_w = OSET_w - \{k\}$
    If $OSET_w = \emptyset$ then do get out of While loop
    { Initialization for the LOADING procedure }
    $j = j + 1$
end

****** END of CASE 4: **********

End { While $OSET_w \neq \emptyset$ }

Figure 9. Main Program

4. EXPERIMENTAL DESIGN

The foregoing heuristic algorithm was coded in Pascal for an evaluation using various input data and run on a micro-computer system. The generation of data will be described later. The Pascal language was selected for the development of a prototype program because it has four advantages over other high-level programming language such as Fortran and BASIC. Those four points are as follows:

1. Rich in data structure.

2. Easily interfaced with the dBASE software package through, for example, a package called dBASE tools from Ashton-Tate.

3. Many algorithms developed for order picking system are written in Pascal so that researchers can reduce time in evaluating algorithms or interfacing them.

4. Enforce a structured programming style.

The prototype program is run on a micro computer because its hardware is affordable and more flexible in terms of time and space. Also Bozer and Goetschalckx [4] in their study concluded that running an algorithm on a micro computer is more cost effective from the user point of view.

Experimental data generation

The data sets used for the experimental study were generated using various distributions such as the normal, uniform, beta, and

gamma. The experiment is a discrete, terminating simulation. By terminating, we mean that the simulation run will be terminated whenever all data, in this study, predetermined number of customer orders have been exhausted. A terminating simulation is conducted, instead of a non-terminating simulation, because no steady-state parameters, for example, queue length, are investigated. The behaviour of a queue length is not studied in this simulation because the characteristics of the proposed algorithm are not related to the queue length. The queue length is more related to the scheduling aspect of both retrieval and storage transactions.

The statistics under investigation are the number of picking lots in any time window, percentage utilization of an order picking vehicle, and execution time in CPU seconds for the operating rules proposed by both Barrett and the author.

The following will give a range of parameter values for distribution functions used to generate the input data file. The input data file consists of two major files: item master and customer orders.

Item Master File

The item master file contains location number, item number, and fractional value for an item number. In this simulation, item number and location number, for data base simplicity, are assumed to be the same. A warehouse in this investigation is assumed to have 100 locations and its location numbers are sequentially numbered from 1 to 100. Then, each location is assigned a fractional value. This

fractional value is generated using a Beta distribution having the form:

$$f(x) = \frac{\Gamma(\theta + \phi)}{\Gamma(\theta)\,\Gamma(\phi)} \; x^{(\theta - 1)} (1 - x)^{(\phi - 1)}$$

for all $0 < x < 1$;

where θ and ϕ values vary the mean, variance, and skewness of the Beta distribution. The θ and ϕ values used are: 10.0 and 1.25, respectively.

Customer Orders File

The customer order file contains item numbers and correponding quantities. Prior to the generation of the customer order file, the total number of orders in any pick wave are generated , apriori, using five different values : 5, 10, 15, 20, 25. For each order in any pick wave, an item number is generated using a uniform distribution with minimum and maximum of 1 to 100, respectively. Its corresponding quantity is generated using a normal distribution with a mean value of four and a variance of four.

The seed number for the pseudo random number generator is different for each simulation run. This will preserve independent behaviour among simulation runs. The pseudo random number generator available with Turbo Pascal is used.

The experimental data is divided into two categories. First, the data will be used to compare Barrett's approach with the proposed algorithm. In this category, due to the nature of Barrett's

algorithm assumptions, the total required capacity of an order is restricted to a maximum value of one (normalised value). In the second category, each order's total required capacity is not forced to be less than or equal to one.

The simulation results are tabulated and graphically displayed using the statistical data derived from the simulation runs.

5. DISCUSSION OF RESULTS

From table 1, the proposed algorithm takes equal or more time to generate results than Barrett's algorithm. And as either the pick wave size or the number of items in an order increases, the difference in CPU seconds of both algorithms widens. This behavior is shown in Figure 10. The CPU seconds increases in both algorithms when the pick wave size increases. The magnitude of this increase is amplified with a larger number of items in an order contained in a pick wave. The smallest difference between those two algorithms occur in pick wave size of five and the difference is basically zero. In contrast, in pick wave sizes of 25, 25 items per order, the number of CPU seconds of the proposed algorithm is almost 4 times as much as that of Barrett's algorithm.

As in Figure 11, the proposed algorithm produces a smaller number of picking lots than does Barrett's algorithm, especially when there is a higher number of items per order. When there is a smaller number of items per order (for example, five items) both algorithms require the same number of picking lots. This characteristic is also true for an increase in pick wave size. In table 1, as the number of items per order increases, the proposed algorithm finds the minimum number of lots more often than does Barrett's algorithm. The minimum number of lots is defined as an integer value that is greater or equal to the total required capacity in a pick wave rounded up to the nearest integer.

As can be seen in table 1, the vehicle utilization value of the proposed algorithm is higher than that of Barrett's algorithm. Since only one pick wave is considered at any one time, the average calculated values seem to be biased on the low side by not considering filling the last vehicle with a portion of the next pick wave. However, in practice, this would not be a problem because all pick waves will be processed together by the algorithm over the entire planning period.

The proposed algorithm was studied under different pick wave sizes and number of items per order. As depicted in Figure 12, the computational time, measured in CPU seconds, of the algorithm requires more time as pick wave size or number of items in an order increases.

Similar to the behaviour of the proposed algorithm with respect to CPU seconds, an increase in either the number of items per order or the pick wave size results in an increase in the number of picking lots or tours formed. This can also be seen in Figure 13.

PICK WAVE SIZE	ITEMS PER ORDER	MINIMUM NUMBER OF LOTS	BARRETT'S ALGORITHM CPU SECONDS	BARRETT'S ALGORITHM NUMBER OF PICKING LOTS	BARRETT'S ALGORITHM VEHICLE UTILIZATION PERCENT	PROPOSED ALGORITHM CPU SECONDS	PROPOSED ALGORITHM NUMBER OF PICKING LOTS	PROPOSED ALGORITHM VEHICLE UTILIZATION PERCENT
5	5	3	0.110	3	79.99	0.110	3	79.99
10	5	5	0.440	5	80.94	0.220	5	80.94
15	5	7	0.220	7	93.24	0.280	7	93.24
20	5	9	0.330	9	91.65	0.380	9	91.65
25	5	12	0.490	12	92.05	0.550	12	92.05
5	10	4	0.110	5	78.30	0.110	4	97.87
10	10	8	0.170	10	78.93	0.330	8	98.67
15	10	14	0.270	15	88.56	0.550	14	94.89
20	10	16	0.440	20	78.42	0.880	16	98.00
25	10	21	0.610	25	81.92	1.210	21	97.52
5	15	4	0.150	5	71.12	0.220	4	88.91
10	15	7	0.170	10	68.79	0.490	8	95.38
15	15	11	0.280	15	67.04	0.820	11	91.43
20	15	14	0.390	20	67.97	1.490	14	97.11
25	15	17	0.550	25	64.23	2.250	17	94.27

Table 1. The results of Barrett's and the proposed algorithms

PICK WAVE SIZE	ITEMS PER ORDER	MINIMUM NUMBER OF LOTS	CPU SECONDS	NUMBER OF PICKING LOTS	VEHICLE UTILIZATION PERCENT
5	5	3	0.110	3	87.35
10	5	5	0.170	5	80.28
15	5	7	0.270	7	92.46
20	5	9	0.380	9	90.80
25	5	11	0.550	11	95.71
5	10	5	0.220	5	84.24
10	10	9	0.330	9	90.16
15	10	13	0.550	13	96.31
20	10	16	0.880	17	93.88
25	10	22	1.150	22	97.81
5	15	7	0.160	7	97.53
10	15	12	0.550	13	92.08
15	15	19	1.100	20	94.89
20	15	25	1.260	25	96.17
25	15	32	1.810	33	96.74
5	20	8	0.220	9	88.66
10	20	17	0.600	17	97.92
15	20	26	0.990	26	97.32
20	20	34	1.590	35	96.08
25	20	42	2.310	43	97.47
5	25	12	0.550	12	92.83
10	25	21	0.710	21	98.34
15	25	31	1.320	32	96.04
20	25	42	2.250	42	98.75
25	25	53	2.800	54	97.45

Table 2. The results of the proposed algorithm using various values of pick wave sizes and number of items per order

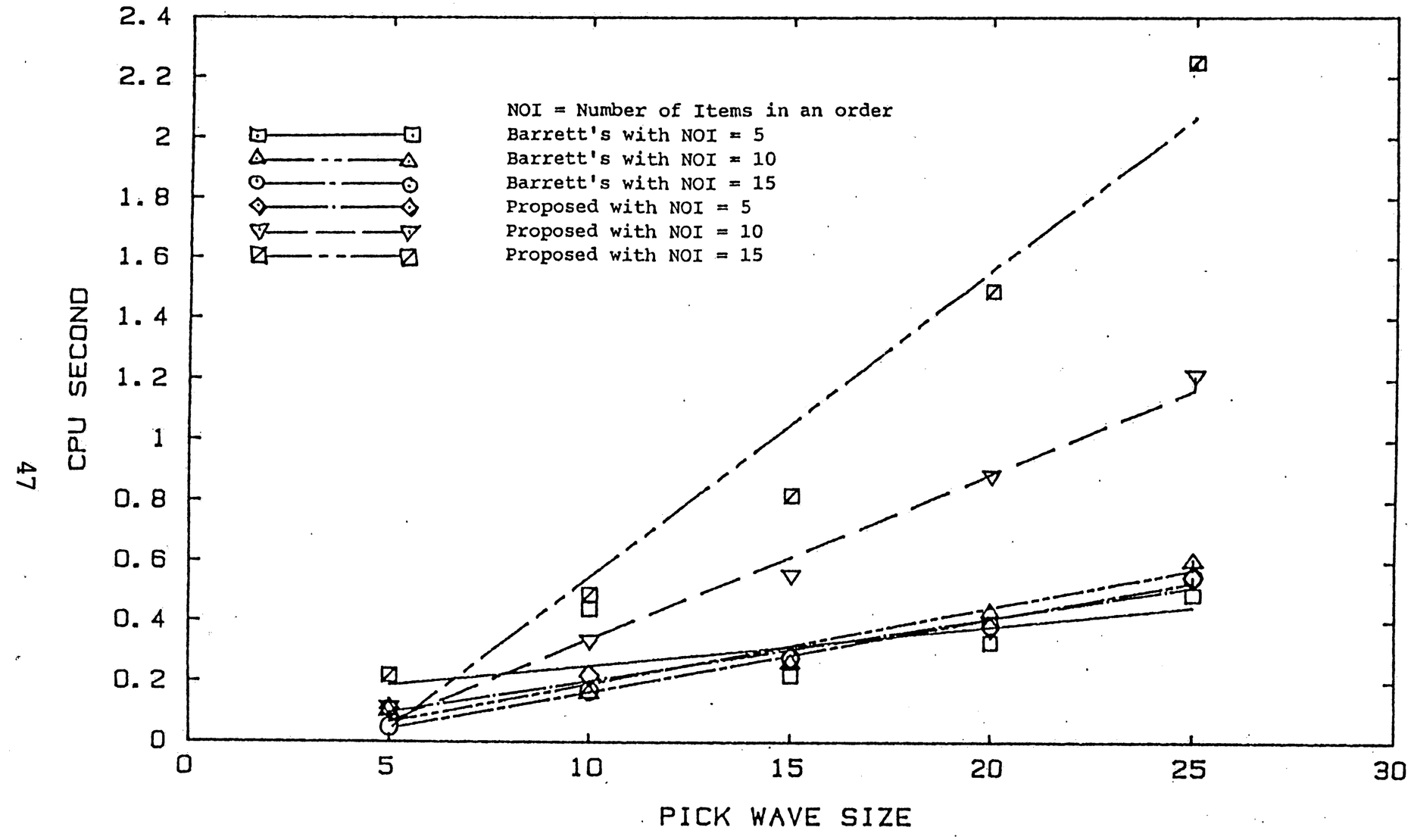


Figure 10. Comparison of Barrett's and Proposed in CPU second

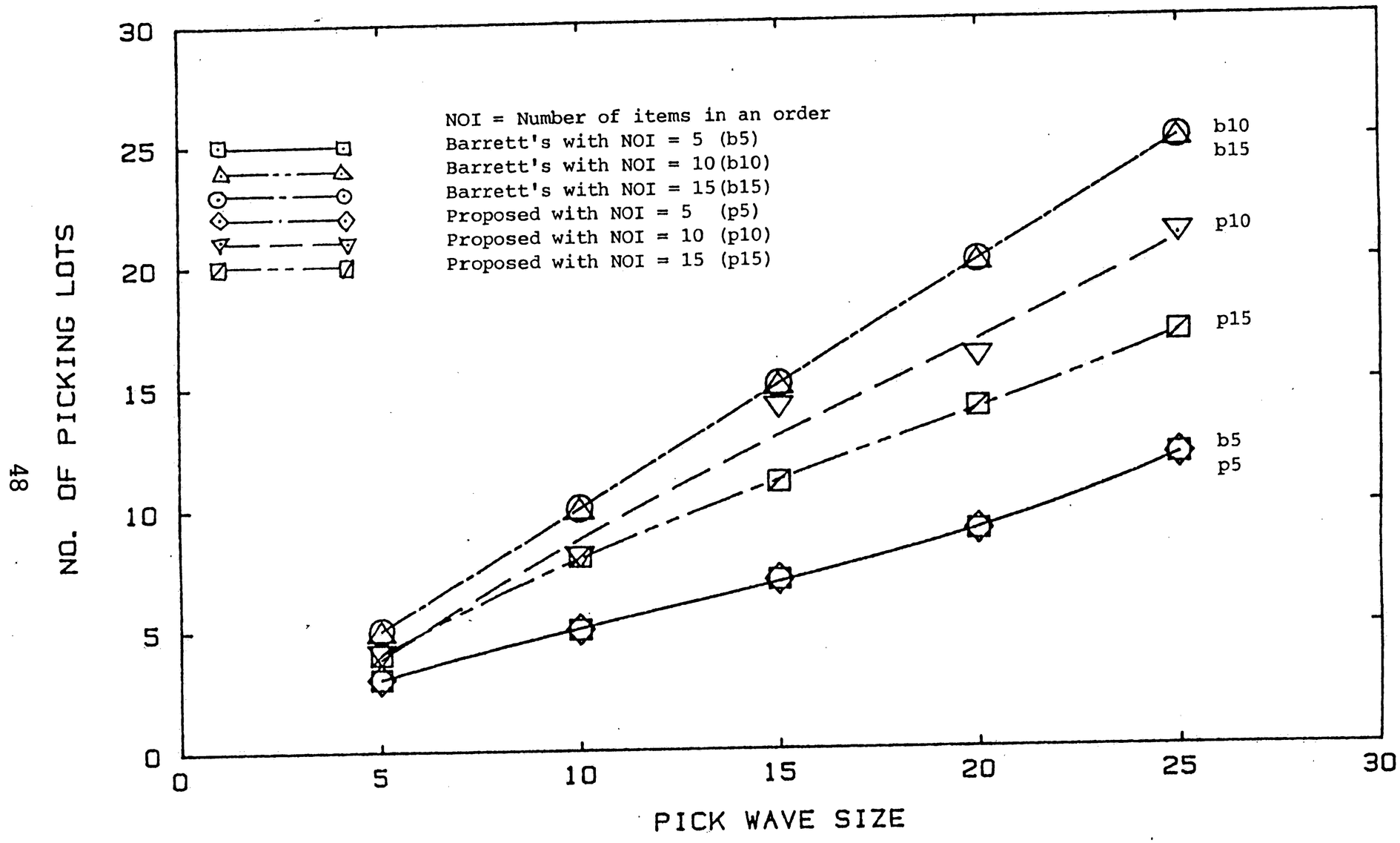


Figure 11. Comparison of Barrett's and Proposed Algorithm in CPU second

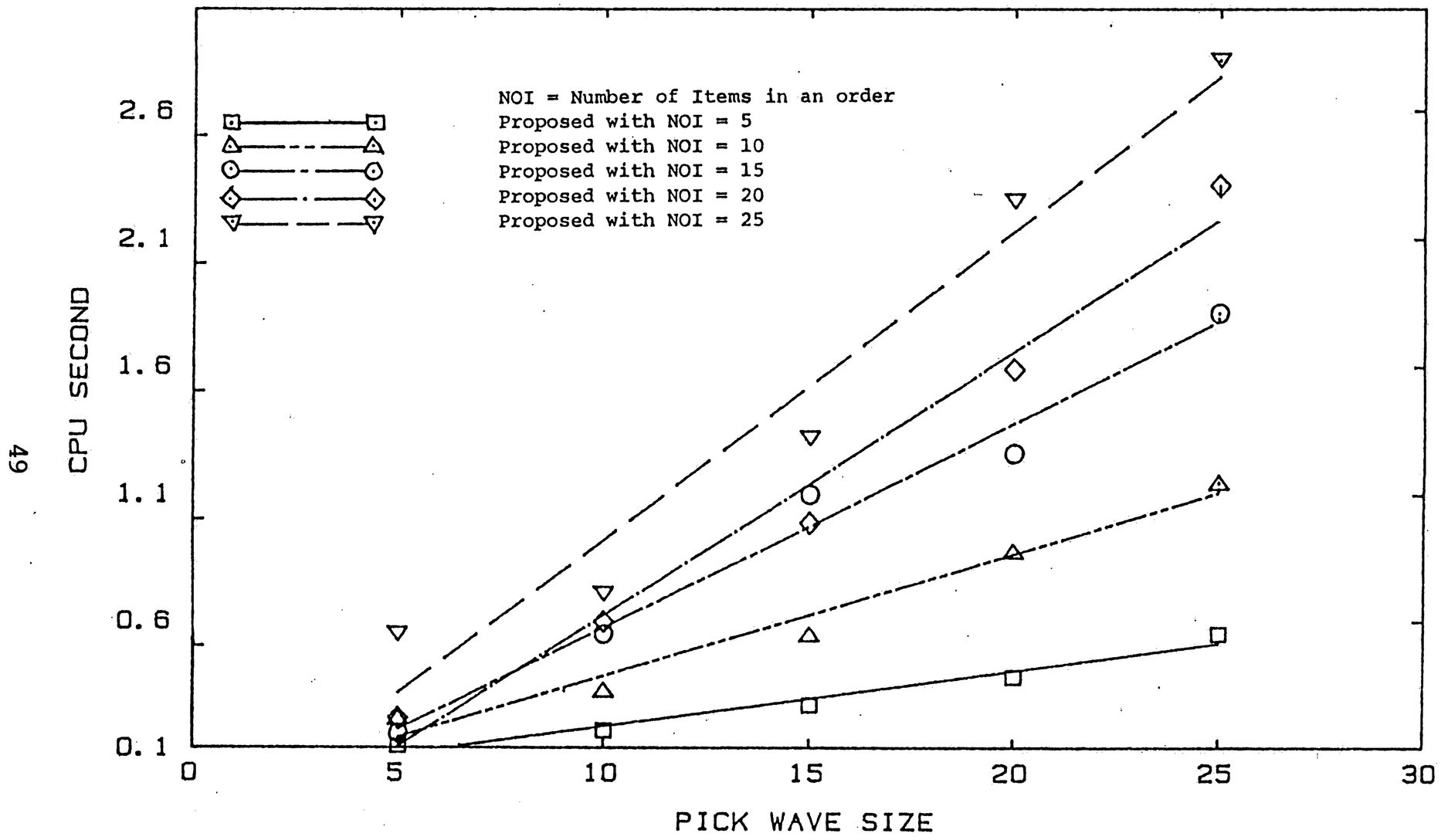


Figure 12. The Behaviour of the Proposed algorithm in CPU second.

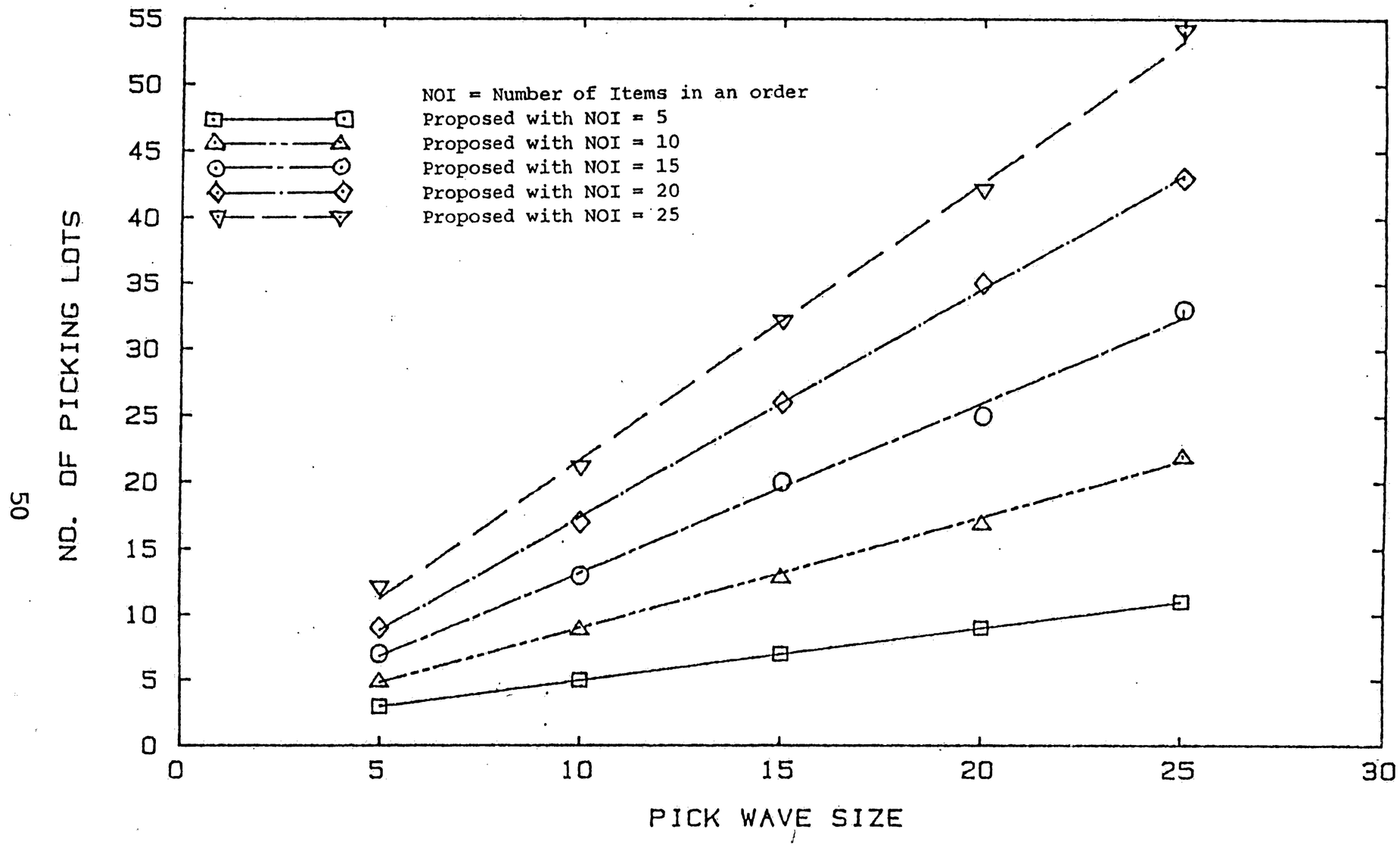


Figure 13. The Behaviour of the Proposed algorithm in Number of Picking Lots

6. CONCLUSIONS

From the discussion of results, the proposed algorithm is superior to Barrett's algorithm with respect to number of picking lots and vehicle utilization percentage. Barrett's algorithm seems to perform well for a pick wave having five orders or less and also each order having five or less items. It is apparent from the simulation that Barrett's algorithm was designed to be used in a non-computerized environment. On the other hand, the proposed algorithm performs well for a pick wave that has more than five orders or orders having more than five items. In addition, this algorithm considers the dimensional sizes of items in a warehouse. This is an improvement over the algorithm proposed by Elsayed and Stern [9].

The proposed algorithm is appropiate for a computerized warehouse. And it is also applicable to an order picking system which has a man-on-board order picking system and also zone picking.

7. FUTURE RESEARCH

New research areas related to the previous model can be generated. The following section will identify four possible research areas that are directly related to this study.

In the model formulation, the optimal number of order picking vehicles required per unit time is not determined. In addition, it appears that little research has been done in this area. Thus, determining a required number of order picking vehicles per shift in an order picking system may be an area of interest for further study. The interrelationship between a number of vehicles and demand distribution could be incorporated into a more comprehensive study of what parameters directly affect the number of vehicles required.

One of the assumptions of the foregoing model is that a replenishment scheduling system is not considered. I feel a simulation study of traffic congestion affected by order picking vehicles and replenishment vehicles would be an interesting research topic. This research will be enhanced if data is collected from a real working warehouse. By doing this a simulation model verification can be easily carried out.

Expert systems have been developed for applications in areas as far ranging as medicine to production. Another obvious application would be warehouse operation. For example, developing a well-planned maintenance schedule incorporating the order picking schedule for order picking vehicles may be one topic. A simulation result can be

integrated with an expert system so that a new picking schedule can be generated to reduce possible congestion indicated by the simulation result.

Also as mentioned earlier, the heuristic traveling salesman algorithm based upon Spacefilling Curves developed by Bartholdi and Platzman [2] can be applied to a rectangular warehouse such as one described in this paper. By incorporating one of the TSP methods with the proposed algorithm, throughput in terms of picking can be studied effectively.

The previously mentioned areas for further research are not exhaustive. They are mentioned because they are related to and are a natural extension of this study.

APPENDIX A

SAMPLE CALCULATIONS

The following is a sample problem and calculations using the algorithm outlined in Figure 3 to Figure 9. A warehouse is assumed to contain ten items and each item has its own pre-assigned location. In this sample problem, location number is the same as item number. The fractional value of all items is assumed to be 0.05. The following are data for three orders in a pick wave.

Order No. (k)	$OCAP_k$	$TNIO_k$	Item Number	Quantity
1	1.25	10	1	3
			2	2
			3	4
			4	2
			5	2
			6	3
			7	2
			8	1
			9	3
			10	3
2	0.30	3	2	2
			4	1
			6	3
3	0.35	4	4	3
			6	1
			7	2
			10	1

From the Main Program shown in Figure 9, a FINDING-SEED-ORDER procedure, shown in Figure 3, is executed.

$$X^* = \{ x \mid MAX [TNIO_k], \forall k \in OSET_w \}$$

$OSET_w$ = { 1,2,3 }

X^* = { x | MAX [10,3,4] } = { 10 }

$TNIO_1$ = 10; therefore, k = 1

k = 1

j = 1

LOADING procedure is executed

working with a new order

n = 1

$VCAP_1$ = 0

SNS_1 = $TNIO_1$ = 10

$VCAP_k$ < C and n <= SNS_1

0 < 1 and 1 <= 10 =====> true

m = n = 1

r = ID_{km} = ID_{11} = 1

PS_{11} = ID_{11} = 1

PL_{11} = LOC_1 = 1

PQ_{11} = QTY_{11} = 3

$VCAP_1$ = $VCAP_1$ + QTY_{11} * FV_1

$VCAP_1$ = 0 + 3 * 0.05 = 0.15

n = n + 1 = 1 + 1 = 2

$VCAP_1$ < C and n <= SNS_1

0.15 < 1 and 2 <= 10 =====> true

m = n = 2

r = ID_{12} = 2

$PS_{12} = ID_{12} = 2$

$PL_{12} = LOC_2 = 2$

$PQ_{12} = QTY_{12} = 2$

$VCAP_1 = 0.15 + 2 * 0.05 = 0.25$

$n = 2 + 1 = 3$

.

.

.

This continues until item number 9

$PS_{19} = ID_{19} = 9$

$PL_{19} = LOC_9 = 9$

$PQ_{19} = QTY_{19} = 3$

$VCAP_1 = 0.95 + 3 * 0.05 = 1.1$

$n = 9 + 1 = 10$

$VCAP_1 < C$ and $n <= SNS_j$

$1.1 < 1$ and $10 <= 10$ =====> false

RETURN to main program, as in Figure 9.

CASE 2 is executed since $VCAP_1 > C$ =====> $1.1 > 1.0$

$n = n - 1 = 10 - 1 = 9$

working with a new order, $m = n = 9$

$SN = m = 9$

$VCAP_1 = VCAP_1 - QTY_{19} * FV_9$

$VCAP_1 = 1.1 - 3 * 0.05 = 0.95$

ADJUSTLOT procedure executed

m = SN = 9

EL = 1 - 0.95 = 0.05

QI = TRUNC [0.05/0.05] = 1

QI > 0 =====> true

$PS_{19} = ID_{km} = 9$

$PL_{19} = LOC_9 = 9$

$PQ_{19} = QI = 1$

$VCAP_1 = 0.95 + 1 * 0.05 = 1.00$

$UTIL\% = VCAP_1 * 100 = 1.00 * 100 = 100\ \%$

$QTY_{19} = QTY_{19} - QI = 3 - 1 = 2$

REINDEXING procedure called

For m = 9 to 10

$ID_{1[(9-9)+1]} = ID_{11} = ID_{19} = 9$

$QTY_{1[(9-9)+1]} = QTY_{11} = QTY_{19} = 2$

$ID_{1[(10-9)+1]} = ID_{12} = ID_{110} = 10$

$QTY_{1[(10-9)+1]} = QTY_{12} = QTY_{110} = 3$

Out of "m" loop

$RN = TNIO_1 - SN + 2 = 10 - 9 + 2 = 3$

For m = 3 to 10

$ID_{1m} = 0$

$QTY_{1m} = 0$

Out of "m" loop

$TNIO_1 = RN - 1 = 3 - 1 = 2$

RETURN to ADJUSTLOT procedure

j = j + 1 = 1 + 1 = 2 (a new picking lot number)

$VCAP_2 = 0$

np = 1

LOADING procedure is executed

working with load remaining, n = np = 1

$SNS_2 = TNIO_1 + (np - 1) = 2 + (1 - 1) = 2$

$VCAP_2 < 1$ and n <= 2

0 < 1 and 1 <= 2 =====> true

m = n - (np - 1) = 1 - (1 - 1) = 1

$PS_{21} = ID_{km} = ID_{11} = 9$

$PL_{21} = LOC_9 = 9$

$PQ_{21} = QTY_{11} = 2$

$VCAP_2 = 0 + 2 * 0.05 = 0.1$

n = n + 1 = 1 + 1 = 2

$VCAP_2 < C$ and n <= SNS_2

0.1 < 1 and 2 <= 2 =====> true

m = n - (np - 1) = 2 - (1 - 1) = 2

$PS_{22} = ID_{km} = ID_{12} = 10$

$PL_{22} = LOC_{10} = 10$

$PQ_{22} = QTY_{km} = QTY_{12} = 3$

$VCAP_2 = 0.1 + 3 * 0.05 = 0.25$

n = n + 1 = 2 + 1 = 3

$VCAP_2 < C$ and n <= SNS_2

0.25 < 1 and 3 <= 2 =====> false

RETURN to Main program

$VCAP_2 < C$ and $n > SNS_2$

$0.25 < 1$ and $3 > 2$ =====> true. ∴ Case 1 is executed

$b = 1$

$OSET_w = OSET_w - [b] = \{ 1,2,3 \} - \{1\} = \{ 2,3 \}$

$OSET_w \neq \emptyset$

ORDER CONGRUENCY procedure called and executed.

$z = 0$

$k = \{ 2,3 \}$ with $b = 1$

for $k = 2$

$MATCH_2 = 0$

For counter = 1 to $TNIO_1$ where $TNIO_1 = 10$

For m = 1 to $TNIO_2$ where $TNIO_2 = 4$

$MATCH_2 = 3$

If $MATCH_2 > 0$, $z = z + 1 = 1 + 0 = 1$

$MI_1 = 2$

for $k = 3$, $MATCH_3 = 4$

$MATCH_3 > 0$, $z = z + 1 = 1 + 1 = 2$

$MI_2 = 3$

If $z > 1$

$X^* = \{ x \mid MAX [3,4] \}$

$X^* = 4$ =====> order no = 3

$np = n = 3$

LOADING procedure executed

working with load remaining, $n = np = 3$

$SNS_2 = TNIO_3 + (np - 1) = 4 + (3 - 1) = 6$

$VCAP_2$ < 1 and n <= 6 =====> true

working with load remaining, m = n - (np - 1)

= 3 - (3 -1) = 1

$PS_{23} = ID_{31} = 4$

$PL_{23} = LOC_4 = 4$

$PQ_{23} = QTY_{31} = 3$

$VCAP_2 = 0.25 + 3 * 0.05 = 0.4$

n = n + 1 = 3 + 1 = 4

Keep on loading until order no. 3 is completely loaded.

$VCAP_2 = 0.6$

n = 7

RETURN to Main program

$VCAP_2$ < C and n > SNS_2

0.6 < 1 and 7 > 6 ===== true. ∴ Case 1 is executed

b = 3

$OSET_W$ = { 2,3 } - { 3 } = { 2 }

np = n = 7

LOADING procedure executed

working with load remaining, n = np = 7

$SNS_2 = TNIO_2 + (np - 1) = 3 + (7 - 1) = 9$

$VCAP_2$ < 1 and n <= 9 =====> true

$PS_{27} = ID_{21} = 2$

$PL_{27} = LOC_2 = 2$

$PQ_{27} = QTY_{21} = 2$

$n = n + 1 = 7 + 1 = 8$

$VCAP_2 = 0.6 + 2 * 0.05 = 0.7$

Load until order no. 2 is completely loaded.

$VCAP_2 = 0.9$

$n = 10$

RETURN to Main program

$VCAP_2 < C$ and $n > SNS_2$

$0.9 < 1$ and $10 > 9$ =====> true. ∴ Case 1 is executed again.

$b = 2$

$OSET_w = \{ 2 \} - \{ 2 \} = \emptyset$

$OSET_w = \emptyset$ =====> end of execution.

Summary of the results :

$PS_{11} = 1;$	$PL_{11} = 1;$	$PQ_{11} = 3;$
$PS_{12} = 2;$	$PL_{12} = 2;$	$PQ_{12} = 2;$
$PS_{13} = 3;$	$PL_{13} = 3;$	$PQ_{13} = 4;$
$PS_{14} = 4;$	$PL_{14} = 4;$	$PQ_{14} = 2;$
$PS_{15} = 5;$	$PL_{15} = 5;$	$PQ_{15} = 2;$
$PS_{16} = 6;$	$PL_{16} = 6;$	$PQ_{16} = 3;$
$PS_{17} = 7;$	$PL_{17} = 7;$	$PQ_{17} = 2;$
$PS_{18} = 8;$	$PL_{18} = 8;$	$PQ_{18} = 1;$
$PS_{19} = 9;$	$PL_{19} = 9;$	$PQ_{19} = 1;$
$VCAP_1 = 1.00;$	$UTIL\%_1 = 100\ \%;$	

$PS_{21} = 9;$	$PL_{21} = 9;$	$PQ_{21} = 2;$
$PS_{22} = 10;$	$PL_{22} = 10;$	$PQ_{22} = 3;$
$PS_{23} = 4;$	$PL_{23} = 4;$	$PQ_{23} = 1;$
$PS_{24} = 6;$	$PL_{24} = 6;$	$PQ_{24} = 1;$
$PS_{25} = 7;$	$PL_{25} = 7;$	$PQ_{25} = 2;$
$PS_{26} = 10;$	$PL_{26} = 10;$	$PQ_{26} = 1;$
$PS_{27} = 2;$	$PL_{27} = 2;$	$PQ_{27} = 2;$
$PS_{28} = 4;$	$PL_{28} = 4;$	$PQ_{28} = 1;$
$PS_{29} = 6;$	$PL_{29} = 6;$	$PQ_{29} = 3;$
$VCAP_2 = 0.90;$	$UTIL\% = 90\ \%;$	

Note : In the implementation of the program, a module should be added to group PS_{jn} having the same value such as PS_{22} and PS_{26}.

BIBLIOGRAPHY

[1] Barrett, Bruce G. *An Empirical Comparison of High-Rise Warehouse Policies for Operator-Controlled Stacker Cranes.* Logistics Research & Analysis, Eastman Kodak Company, May 1977.

[2] Bartholdi, J. J., and Platzman, L. K. *Design of Efficient Bin-Numbering Schemes for Warehouses.* Submitted to Material Flow, September 1985.

[3] Bentley, Jon Louis. *A Case Study in Applied Algorithm Design.* Computer, Vol 17, No. 2, 1984, pp. 75-88.

[4] Bozer, Yavuz A. and Goetschalckx, M. P. *Relative Performance of Micro and Mainframe Computers.* Computers and Industrial Engineering, Vol. 10, No. 2, 1986, pp. 117-119.

[5] Bozer, Yavuz A., Schorn, Ellen C., and Sharp, Gunter P. *Geometric Approaches to Solve The Chebyshev Travelling Salesman Problem.* Currently under review for IIE Transactions.

[6] Bozer, Yavuz A., and White, John A. *Travel-Time Models for Automated Storage/Retrieval Systems.* IIE Transactions, Vol. 16, No. 4, 1984, pp. 329-338.

[7] Eilon, Samuel, and Christofides, Nicos. *The Loading Problem.* Management Science, Vol. 17, No. 5 (January 1971).

[8] Elsayed, E. A. Design and Scheduling Rules of Automated Storage/Retrieval Systems. Rutgers University, Industrial Engineering Department, working paper 85-112, 1985.

[9] Elsayed, E. A., and Stern, Richard G. *Computerized Algorithms for Order Processing in Automated Warehousing Systems.* International Journal of Production Research, Vol. 21, No. 4, 1983, pp. 579-586.

[10] Gillett, Billy E., and Miller, Leland R. *A Heuristic Algorithm for the Vehicle-Dispatch Problem.* Operations Research, Vol. 22, No. 2, 1974, pp. 340-349.

[11] Golden, B., Bodin, L., Doyle, T., and Stewart, W. Jr. *Approximate Traveling Salesman Algorithms.* Operations Research, Vol. 28, No. 3, 1980, pp. 694-711.

[12] Graves, Stephen C., Hausman, Warren H., and Schwarz, Leroy B. *Storage-Retrieval Interleaving in Automatic Warehousing Systems*. Management Science, Vol 23, No. 9, 1977, pp. 935-945.

[13] Gudehus, T. *Principles of Order Picking: Operations in Distribution and Warehousing Systems*. W. Giradet, Essen, West Germany, 1973. (in German language)

[14] Hausman, W. H., Schwarz, L. B., and Graves, S. C. *Optimal Storage Assignment in Automatic Warehousing Systems*. Management Science, Vol. 22, No. 6, Feb 1976, pp. 629-638.

[15] Karp, R. M., and Held, M. *The Traveling Salesman Problem and Minimum Spanning Trees*. Operations Research, Vol. 18, 1970, pp. 1138-1162.

[16] Lin, Shen, and Kerninghan, B. W. *A Heuristic Algorithm for the Traveling Salesman Problem*. Computer Science Technical Report No. 1, Bell Labs, April 1972.

[17] Mayer, Hugo E. Jr. *Storage and Retrieval of Material*. The Western Electric Engineer, Vol. 5, No. 1, 1961, pp. 42-48.

[18] Papadimitriou, Christos H. *The Euclidean Traveling Salesman Problem is NP-Complete*. Theoretical Computer Science, Vol. 4, No. 3, 1977, pp. 237-244.

[19] Parker, R. G., and Rardin, R. L. *The Traveling Salesman Problem: An Update of Research*. Naval Research Logistics Quarterly, Vol. 30, 1983, pp. 69-96.

[20] Ratliff, H. Donald, and Rosenthal, Arnon S. *Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem*. Operations Research, Vol. 31, No. 3, 1983, pp. 507-521.

[21] Sand, G. M. *Stacker Crane Product Handling System*. Eastman Kodak Company, September 1976.

[22] Hausman, Warren H., Graves, Stephen C., and Schwarz, Leroy B. *Scheduling Policies for Automatic Warehousing Systems: Simulation Results*. IIE Transactions, Vol. 10, No. 3, 1978, pp. 260-270.

VITA

The author was born the son of Ie Yat and Kwong Yoeng Khoe on December 27, 1960 in Jakarta, Indonesia. The author earned a High School Diploma from SMAK-I (also known as Pintu Air), located in Jakarta, in May 1979.

Moved by the spirit of adventure and thirst for education, the author entered Ryerson Polytechnical Institute, Toronto, Ontario, Canada in September 1980 and earned both a Diploma in Mechanical Engineering Technology and a Certificate in Computer Programming Applications in June 1983.

The author then enrolled in the Mechanical Engineering program at the University of Missouri-Rolla (UMR), Rolla, Missouri in August 1983 and graduated Cum Laude with a Bachelor of Science in Mechanical Engineering in December 1984. After graduation, the author accepted a Graduate Teaching Assistantship from UMR to pursue advanced study in Mechanical Engineering from January 1985 to May 1985.

In June 1985 the author accepted a Graduate Research Assistantship from Lehigh University, Bethlehem, Pennsylvania to pursue a Masters degree program in Manufacturing Systems Engineering (MSE).